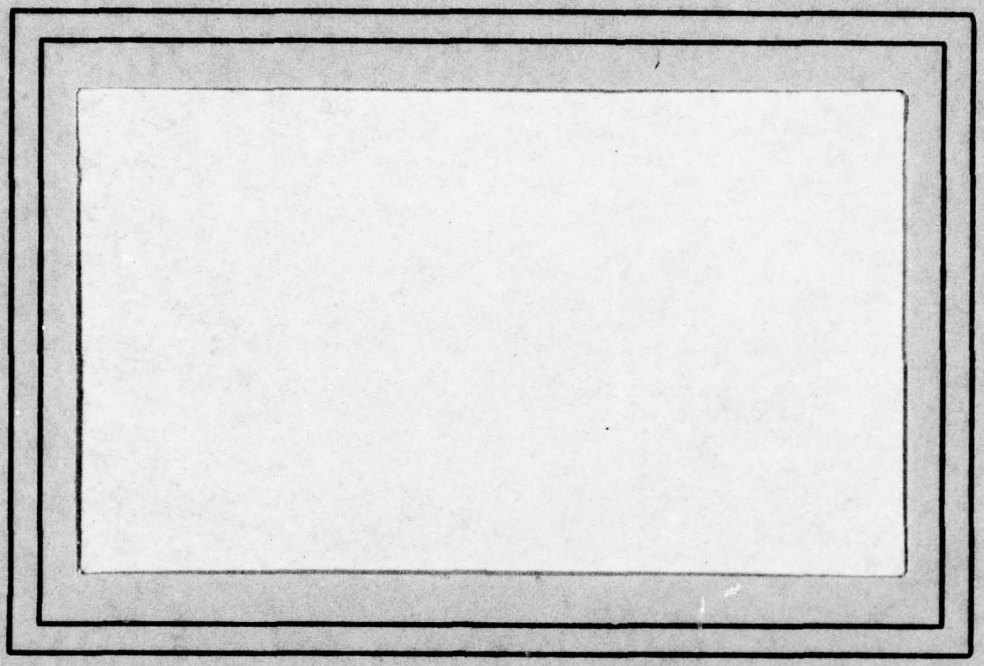
AD-A065 624 MARYLAND UNIV COLLEGE PARK COMPUTER SCIENCE CENTER F/G 9/2
SRRIT - A FORTRAN SUBROUTINE TO CALCULATE THE DOMINANT INVARIAN--ETC(U)
NOV 78 G W STEWART N00014-76-C-0391
UNCLASSIFIED TR-514 NL

1 OF 1
AD A065824

END
DATE FILMED
5-79
DDC

(1) LEVEL II



UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND
20742

LEVEL II

Technical Report TR-514
ONR-N00014-76-C-0391

November 1978

SRRIT - A FORTRAN Subroutine to Calculate the Dominant Invariant Subspaces of a Real Matrix

G. W. Stewart*

53 p.

ABSTRACT

SRRIT is a FORTRAN program to calculate an approximate orthonormal basis for a dominant invariant subspace of a real matrix A. Specifically, given an integer m, SRRIT attempts to compute a matrix Q with m orthonormal columns and real quasi-triangular matrix T of order m such that the equation

$$AQ = QT$$

is satisfied up to a tolerence specified by the user. The eigenvalues of T are approximations to the m largest eigenvalues of A, and the columns of Q span the invariant subspace corresponding to those eigenvalues. SRRIT references A only through a user provided subroutine to form the product AQ; hence it is suitable for large sparse problems.

*This work was supported in part by the Office of Naval Research under Contract No. N 00014-76-C-0391.

SRRIT - A FORTRAN Subroutine to Calculate the Dominant Invariant Subspaces of a Real Matrix

G. W. Stewart

DESCRIPTION

1. Introduction

The program described in this paper is designed primarily to solve eigenvalue problems involving large, sparse, real matrices. The programs attempt to calculate a set of the largest eigenvalues of the matrix in question. In addition they calculate a canonical orthonormal basis for the invariant subspace spanned by the eigenvectors and principal vectors corresponding to the set of eigenvalues. No explicit representation of the matrix is required; instead the user furnishes a subroutine to calculate the product of the matrix with a vector.

Since the programs do not produce a set of eigenvectors corresponding to the eigenvalues computed, it is appropriate to begin with a mathematical description of what is actually computed and how the user may obtain eigenvectors from this output if he so desires. Let A be a matrix of order n with eigenvalues $\lambda_1,\lambda_2,\ldots,\lambda_n$ ordered so that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n| .$$

An invariant subspace of A is any subspace $\mathcal{Q}$ for which

$$x \in \mathcal{Q} \quad \Rightarrow \quad Ax \in \mathcal{Q} ;$$

i.e., the subspace is transformed into itself by the matrix A.

If $\mathcal{Q}$ is an invariant subspace of A and the columns of $Q = (q_1, q_2, \ldots, q_m)$ form a basis for $\mathcal{Q}$, then $Aq_i \in \mathcal{Q}$, and hence Aq_i can be expressed as a linear combination of the columns of Q; i.e., there is an m-vector t_i such that $AQ = Qt_i$. Setting

$$T = (t_1, t_2, \ldots, t_m)$$

we have the relation

$$AQ = QT \,. \tag{1.1}$$

In fact the matrix T is just the representation of the matrix A in the subspace $\mathcal{Q}$ with respect to the basis Q.

If x is an eigenvector of T corresponding to the eigenvalue λ, then it follows from (1.1) and the relation $Tx = \lambda x$ that

$$A(Qx) = \lambda(Qx) \,, \tag{1.2}$$

so that Qx is an eigenvector of A corresponding to the eigenvalue λ. Thus the eigenvalues of T are also eigenvalues of A. Conversely if $\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_m}$ are any m eigenvalues of A that are distinct from the other n-m eigenvalues, then there is a unique invariant subspace of dimension m corresponding to these eigenvalues; i.e., the eigenvalues of T in (1.1) are precisely $\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_m}$.

If $|\lambda_i| > |\lambda_{i+1}|$, then there is a unique <u>dominant invariant subspace</u> $\mathcal{Q}_i$ corresponding to $\lambda_1, \lambda_2, \ldots, \lambda_i$. When $\mathcal{Q}_i$ and $\mathcal{Q}_{i+1}$ exist, $\mathcal{Q}_i \subset \mathcal{Q}_{i+1}$.

The subroutine SRRIT attempts to compute a nested sequence of orthonormal bases of $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_m$. Specifically, if all goes well, the subroutine produces a matrix Q with orthonormal columns having the property that if $|\lambda_i| > |\lambda_{i+1}|$ then $q_1, q_2, \ldots, q_i$ span $\mathcal{Q}_i$.

The case where λ_{i-1} and λ_i are a complex conjugate pair, and hence $|\lambda_{i-1}| = |\lambda_i|$, is treated as follows. The matrix Q is calculated so that the matrix T in (1.1) is quasi-triangular; i.e., T is block triangular with 1×1 and 2×2 blocks on its diagonal. The structure of a typical quasi-triangular matrix is illustrated below for $m = 6$:

$$\begin{pmatrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & x \end{pmatrix}.$$

The 1×1 blocks of T contain the real eigenvalues of A and the 2×2 blocks contain conjugate pairs of complex eigenvalues. This arrangement enables us to work entirely with real numbers, even when some of the eigenvalues of T are complex. The existence of such a decomposition is a consequence of Schur's theorem (see [9]).

The eigenvalues of the matrix T computed by the program appear in descending order of magnitude along its diagonal. For fixed i let $Q^{[i]} = (q_1, q_2, \ldots, q_i)$ and let $T^{[i]}$ be the leading principal submatrix of T of order i. Then if the i-th diagonal entry of T does not begin a 2×2

block, we have

$$AQ^{|i} = Q^{|i}T^{i} .$$

Thus the first i columns of Q span the invariant subspace corresponding to the first i eigenvalues of T. When $|\lambda_i| > |\lambda_{i+1}|$ this is the unique dominant invariant subspace $\mathcal{Q}_i$. When $|\lambda_i| = |\lambda_{i+1}|$ the columns of $Q^{|i}$ span a dominant invariant subspace; but is is not unique, since there is no telling which comes first, λ_i or λ_{i+1}.

Any manipulations of A within the subspace $\mathcal{Q}$ corresponding to Q can be accomplished by manipulating the matrix T. For example,

$$A^k Q = QT^k ,$$

so that if $f(A)$ is any function defined by a power series, we have

$$f(A)Q = Qf(T) .$$

If the spectrum of A that is not associated with Q is negligible, considerable work can be saved by working with the generally much smaller matrix T in the coordinate system defined by Q. If explicit eigenvectors are desired, they may be obtained by evaluating the eigenvectors of T and applying (1.2). The programs hqr2 in [12] and HQR1 in [7] will evaluate the eigenvectors of a quasi-triangular matrix.

2. Usage

SRRIT is a FORTRAN subroutine to calculate the basis for $\mathcal{Q}_m$ described in Section 1. The calling sequence for SFFIT is

```
CALL SRRIT(Q,AQ,ATQ,N,NV,M,EPS,MAXIT,START,T,ER,EI,
    TYPE,RSD,RSDX)
```

with (starred parameters are altered by the subroutine)

*Q(N,M)	A real array that on return contains the approximation to Q. Initially Q may contain a starting approximation (cf. START).
*AQ(N,M)	A real array that on return contains the product AQ.
ATQ	The name of a FORTRAN subroutine that computes the product AQ. For details see below.
N	The order of A.
*NV	The number of vectors to compute. On return, NV contains the number of columns of Q that have converged.
M	The number of columns of Q. M must be greater than or equal to NV.
EPS	A convergence criterion.
MAXIT	An integer containing the maximum number of iterations to perform.
START	An integer that tells the initial status of Q. If START < 0, a starting approximation is to be generated randomly. If START ≥ 0, then Q initially contains a starting approximation; and if START ≥ 1, then the columns of Q are assumed to be orthonormal.
*T(M,M)	A real array that on return contains the approximation to the matrix T of (1.1).
*ER(M)	A real array that on return contains the real parts of the eigenvalues of T.
*EI(M)	A real array that on return contains the imaginary parts of the eigenvalues of T.

*TYPE(M) An integer array whose i-th entry on return is

0 if the i-th eigenvalue is real
1 if the i-th eigenvalue is the first of a conjugate pair of complex eigenvalues
2 if the i-th eigenvalue is the second of a conjugate pair of complex eigenvalues
-1 if the i-th eigenvalue was not correctly determined

*RSD(M) A real array whose i-th entry is the 2-norm of the residual associated with the i-th column of Q [cf. (3.2) below].

*RSDX(M) An integer array whose i-th entry is the iteration at which the i-th entry of RSD was computed.

The dimensions in the parameter descriptions are the smallest for which the program will work. In the program listed here they are set for values of N up to five hundred and M up to ten. To accommodate larger problems, change the dimension 500 to the largest expected value of N and the dimension 10 to the largest expected value of M throughout SRRIT and its auxiliary subroutines (n.b. this includes the dimension information in the subroutine calls in SRRSTP).

The user may furnish a starting approximation to the matrix Q in the array Q. Actually all that is required is a set of vectors whose column space approximates Q_m. If such a starting approximation is furnished, the parameter START should be set greater than or equal to zero. If the starting vectors are orthonormal, the parameter START should be set positive. If START is negative, Q is initialized with random numbers and orthogonalized to provide the starting approximation.

The user is required to furnish a subroutine to calculate the product AQ. The calling sequence for this subroutine is

CALL ATQ(Q,AQ,L1,L2)

with

Q(N,M)	A real array containing the matrix Q.
AQ(N,M)	A real array. On return columns L1 through L2 of AQ should contain the product of the matrix A with columns L1 through L2 of Q.
L1, L2	Integers which specify which columns of Q to multiply by the matrix A.

A call to ATQ causes the iteration counter to be increased by one, so that the parameter MAXIT is effectively a limit on the number of calls to ATQ.

The convergence criterion is described in detail in Sections 3 and 4. Essentially the matrices Q and T calculated by the program will satisfy

$$(A+E)Q^{|NV} = Q^{|NV}T^{|\widetilde{NV}} \tag{2.1}$$

where NV (on return) is the number of columns that have converged and E is of order EPS. From this it is seen that EPS should be small compared with A. The criterion insures that the well-conditioned eigenvalues of A will be calculated accurately, and the well-conditioned eigenvectors can be calculated accurately from Q and T.

The rate of convergence of the i-th column of Q depends on the ratio $|\lambda_{M+1}/\lambda_i|$. For this reason it may be desirable to take the number of columns M of Q to be greater than the number of columns NV that one desires to compute. For example, if the eigenvalues of A are 1.0, 0.9, 0.5,... it will pay to take M = 2, even if only the eigenvector corresponding to 1.0 is desired.

Since SRRIT is designed primarily to calculate the largest eigenvalues of a large matrix, no provisions have been made to handle zero eigenvalues. In particular, zero eigenvalues can cause the program to stop in the auxiliary subroutine ØRTH.

SRRIT is supported by a number of auxiliary subroutines (SRRSTP, RESID, GRØUP, ØRTH, CØND, RANDØM) which are described in Section 5. It also requires the EISPACK subroutines ØRTHES and ØRTRAN [7], and the subroutines HQR3, EXCHNG, SPLIT, and QRSTEP [11].

SRRIT can be used as a black box. As such the first NV vectors it returns will satisfy (2.1), although not as many vectors as the user requests need have converged by the time MAXIT is reached. However, the construction of the program has involved a number of arbitrary decisions. Although the author has attempted to make such decisions in a reasonable manner, it is too much to expect that the program will perform efficiently on all distributions of eigenvalues. Consequently the program has been written in such a way that it can be easily modified by someone who is familiar with its details. The purpose of the next three sections is to provide the interested user with these details.

3. Method

The Schur vectors Q of A are computed by a variant of simultaneous iteration, which is a generalization of the power method for finding the dominant eigenvector of a matrix. The method has an extensive literature [1,2,3,5,8], and Rutishauser [6] has published a program for symmetric matrices from which many of the devices in SRRIT have been drawn. The method about to be described has been analyzed in [10].

The iteration for computing Q may be described briefly as follows. Start with an $n \times m$ matrix Q_0 having orthonormal columns. Given Q_ν, form $Q_{\nu+1}$ according to the formula

$$Q_{\nu+1} = (AQ_\nu)R_{\nu+1}^{-1} ,$$

where $R_{\nu+1}$ is either an identity matrix or an upper triangular matrix chosen to make the columns of $Q_{\nu+1}$ orthonormal (just how often such an orthogonalization should be performed will be discussed below). If $|\lambda_m| > |\lambda_{m+1}|$, then under mild restrictions on Q_0 the column space of Q_ν approaches Q_m.

The individual columns of Q_ν will in general approach the corresponding columns of the matrix Q defined in Section 1; however the rate of convergence of the i-th column is proportional to $\max \{|\lambda_i/\lambda_{i-1}|^\nu, |\lambda_{i+1}/\lambda_i|^\nu\}$ and may be intolerably slow. The process may be accelerated by the occasional application of a "Schur-Rayleigh-Ritz step" (from which SRRIT derives its name), which will now be described. Start with Q_ν just after an orthogonalization step, so that $Q_\nu^T Q_\nu = I$. Form the matrix

$$B_\nu = Q_\nu^T A Q_\nu,$$

and reduce it to ordered quasi-triangular form T_ν by an orthogonal similarity transformation Y_ν:

$$Y_\nu^T B_\nu Y_\nu = T_\nu . \tag{3.1}$$

Finally overwrite Q_ν with $Q_\nu Y_\nu$.

The matrices Q_ν formed in this way have the following property. If $|\lambda_{i-1}| > |\lambda_i| > |\lambda_{i+1}|$, then under mild restrictions on Q_0 the i-th column $q_i^{(\nu)}$ of Q_ν will converge to the i-th column q_i of Q at a rate proportional to $|\lambda_{m+1}/\lambda_i|^\nu$. Thus not only is the convergence accelerated, but the first columns of Q_ν tend to converge faster than the later ones.

A number of practical questions remain to be answered.

1. How should one determine when a column of Q_ν has converged?
2. Can one take advantage of the early convergence of some of the columns of Q_ν to save computations?
3. How often should one orthogonalize the columns of the Q_ν?
4. How often should one perform the SRR acceleration described above?

Here we shall merely outline the answers to these questions. The details will be given in the discussion of SRRIT.

1. Convergence. If $|\lambda_{i-1}| = |\lambda_i|$ or $|\lambda_i| = |\lambda_{i+1}|$, the i-th column of Q is not uniquely determined; and when $|\lambda_i|$ is close to $|\lambda_{i-1}|$ or $|\lambda_i|$, the i-th column cannot be computed accurately. Thus a convergence criterion based on the i-th column $q_i^{(\nu)}$ of Q_ν becoming stationary is likely to fail when A has equimodular eigenvalues. Accordingly we have

adopted a different criterion which amounts to requiring that the relation (1.1) is almost satisfied. Specifically, let $t_i^{(\nu)}$ denote the i-th column of T_ν in (3.1). Then the i-th column of the Q_ν produced by the SRR step is said to have converged if the 2-norm (see [9] for a definition) of the residual vector

$$r_i^{(\nu)} = Aq_i^{(\nu)} - Q_\nu t_i^{(\nu)} \tag{3.2}$$

is less than some prescribed tolerance.

If this criterion is satisfied for each column of Q_ν, then the residual matrix

$$R_\nu = AQ_\nu - Q_\nu T_\nu$$

will be small. This in turn implies that there is a small matrix $E_\nu = -R_\nu Q_\nu^T$ such that

$$(A+E_\nu)Q_\nu = Q_\nu T_\nu ,$$

so that Q_ν and T_ν are the matrices associated with the slightly perturbed matrix $A + E_\nu$, provided only that some small eigenvalue of $A + E_\nu$ has not by happenstance been included in T_ν. To avoid this possibility we group nearly equimodular eigenvalues together and require that their average value has settled down before testing their residuals. In addition a group of columns is tested only if the preceding columns have all converged.

2. Deflation. The theory of the iteration indicates that the initial columns of the Q_ν will converge before the later ones. When this happens considerable computation can be saved by freezing these columns. This saves

multiplying the frozen columns by A, orthogonalizing them when $R_{\nu+1} \neq I$, and work in the SRR step.

3. Orthogonalization. The orthogonalization of the columns of AQ_ν is a moderately expensive procedure which is to be put off as long as possible. The danger in postponing orthogonalization is that cancellation of significant figures can occur when AQ_ν is finally orthogonalized, as it must be just before an SRR step. In [10] it is shown that one can expect no more than

$$t = k \log_{10} \kappa(T) \tag{3.3}$$

decimal digits to cancel after k iterations without orthogonalization (here $\kappa(T) = \|T\|\|T^{-1}\|$ is the condition number of T with respect to inversion). The relation (3.3) can be used to determine the number of iterations between orthogonalizations.

4. SRR-steps. The SRR-step described above does not actually accelerate the convergence of the Q_ν; rather it unscrambles approximations to the columns of Q_m that are already present in the column space of Q_ν and orders them properly. Therefore, the only time an SRR step needs to be performed is when it is expected that a column has converged. Since it is known from the theory of the iteration that the residuals in (3.2) tend linearly to zero, the iteration at which they will satisfy the convergence criterion can be predicted from their values at two iterations. As with convergence, this prediction is done in groups corresponding to nearly equimodular eigenvalues.

4. Details of SRRIT

In designing SRRIT, we have tried to make it easily modifiable. This has been done in two ways. First, we have defined a number of important control parameters and given them values at the beginning of the program. The knowledgeable user may alter these values to improve the efficiency of the program in solving particular problems. Second, a number of important tasks have been isolated in independent subroutines. This should make it easy to modify the actual structure of SRRIT, should the user decide that such radical measures are necessary. In this section we shall describe SRRIT in some detail, specifying the action of the control parameters. In the next section we shall describe the supporting subroutines.

Here follows a list of the control parameters with their initial values and a brief description of their functions.

INIT (5)	a number of initial iterations to be performed at the outset.
STPFAC (2.0)	a constant used to determine the maximum number of iterations before the next SRR step.
ALPHA (1.0) BETA (1.1)	parameters used in predicting when the next residual will converge.
GRPTOL (0.001)	a tolerance for grouping equimodular eigenvalues
CNVTOL (0.001)	a convergence criterion for the average value of a cluster of equimodular eigenvalues.
ORTTOL (2.0)	the number of decimal digits whose loss can be tolerated in orthogonalization steps.
SEED (69)	a seed for the random number generator that initializes Q.

We now give an informal description of SRRIT as it appears in the ALGORITHM section. The variable L points to the first column of Q that has not converged. The variable IT is the iteration counter. The variable NXTSRR is the iteration at which the next SRR step is to take place, and the variable DORT is the interval between orthogonalizations.

```
srrit:
1.  initialize control parameters
2.  initialize
    1.  IT = 0
    2.  L = 1
    3.  Initialize Q as prescribed by START
3.  srr:  loop
    1.  perform an SRR step
    2.  compute the residuals
    3.  check convergence, resetting L if necessary
    4.  if L > NV or IT ≥ MAXIT then leave srr
    5.  calculate NXTSRR
    6.  calculate DØRT and NXTØRT
    7.  Q = AQ; IT = IT+1
    8.  orth:  loop until IT = NXTSRR
        1.  power:  loop until IT = NXTØRT
            1.  AQ = A*Q
            2.  Q = AQ
            3.  IT = IT+1
        end power
        2.  orthogonalize Q
        3.  NXTØRT = min (NXTSRR,IT+DØRT)
    end orth
end srr
4.  NV = L-1
end srrit
```

The details of this outline are as follows (the numbers correspond to the statements in the algorithm).

2.3. If START < 0, then Q is initialized using the function RANDØM. If START ≤ 0, the columns of Q are orthogonalized by the subroutine ØRTH.

3. This is the main loop of the program. Each time it is executed an SRR step is performed and convergence is tested.

3.1. The SRR step is performed by the subroutine SRRSTP, which returns the new Q and A*Q, as well as T and its eigenvalues.

3.2. The residuals are computed by the subroutine RESID.

3.3. The algorithm for determining convergence is the following. Starting with the L-th eigenvalue, the subroutine GRØUP is called to determine a group of nearly equimodular eigenvalues, as defined by the parameter GRPTØL. The same is done for the old eigenvalues from the last SRR step. If the groups have the same number of eigenvalues and the average value of the eigenvalues has settled down (CNVTØL), then the residuals are averaged and tested against EPS. If the test is successful, L is increased by the number in the group, and the tests are repeated. Otherwise control is passed to statement

3.4. where the two termination conditions for SRRIT are tested.

3.5. The iteration at which the next SRR-step is to take place (NXTSRR) is determined as follows. NXTSRR is tentatively set equal to STPFAC*IT. If the number of eigenvalues in the new and old groups corresponding to the next set of unconverged eigenvalues is the same, the RMS average of the norms of the residuals of each group is calculated (ARSD, AØRSD). If ARSD < EPS, then NXTSRR = IT+1. If ARDS > AØRSD, then NXTSRR = STPFAC*IT. Otherwise

$$NXTSRR = \min (IT+ALPHA+BETA*DSRR, STPFAC*IT)$$

where

$$DSRR = (\text{ØRSDX-RSDX}) \frac{\log (ARSD/EPS)}{\log (ARSD/A\text{ØRSD})} .$$

Finally NXTSRR is constrained to be less than or equal to MAXIT.

3.6. The interval DØRT between orthogonalizations is computed from (3.3):

$$D\text{ØRT} = \max (1, \text{ØRTTØL}/\log_{10} k(T)),$$

where the condition number $k(T)$ is calculated by the function CØND. The next orthogonalization occurs at

$$NXT\text{ØRT} = \min (IT+D\text{ØRT}, NXTSRR) .$$

3.7. Since the SRR step computes a product AQ, the iteration count must be increased and AQ placed back in Q.

3.8. Loop on orthogonalizations.

3.8.1. Loop overwriting Q with the product A*Q.

4. Set NV to the number of vectors that have actually converged and return.

5. Auxiliary Subroutines

In this section we shall describe the subroutines called by SRRIT. Some of these subroutines have been coded in greater generality than is strictly required by SRRIT in order to make the program easily modifiable by the user.

SRRSTP(Q,AQ,ATQ,L,M,N,T,ER,EI,TYPE,ØER,ØEI,ØTYPE) .

This subroutine performs an SRR step on columns L through M of Q. After forming AQ and $T = Q^T(AQ)$, the routine calls ØRTHES, ØRTRAN, and HQR3 to reduce T to ordered quasi-triangular form. The triangularizing transformation is postmultiplied into Q and AQ. The eigenvalues from the last step are stored in the arrays ØER, ØEI and ØTYPE, and the new eigenvalues are placed in the arrays ER, EI, and TYPE.

RESID(Q,AQ,T,RSD,RSDX,ØRSD,ØRSDX,L1,L2,M,N,IT,TYPE) .

This subroutine computes the norm of the residuals (3.2) for columns L1 through L2 of Q. The old residuals and their iteration numbers are saved in the arrays ØRSD and ØRSDX. The I-th entry of the array RSDX is set to IT depending on whether or not TYPE(I) ≥ 0. For a complex pair of eigenvalues, the RMS average of the norms of their two residuals is returned.

GRØUP(ER,EI,TYPE,GRPTØL,L,M,N,NGRP,CTR,AE)

This subroutine locates a group of approximately equimodular eigenvalues $\lambda_L, \lambda_{L+1}, \ldots, \lambda_{L+NGRP-1}$. The eigenvalues so grouped satisfy

$$\left| \lambda_i - CTR \right| \leq GRPTOL*CTR \qquad (i=L,L+1,\ldots,L+NGRP-1) .$$

The mean of the group is returned in AE.

ØRTH(AQ,Q,L,M,N)

For J = L,L+1,...,M this subroutine orthogonalizes the J-th column of AQ with respect to columns 1,2,...,L-1 of Q and columns L,L+1,...,J-1 of AQ. The results are returned in Q. The method used is the modified Gram-Schmidt method with reorgonalization. No more than NTRY reorthogonalizations are performed, after which the routine executes a STØP. The routine will also stop if any column becomes zero.

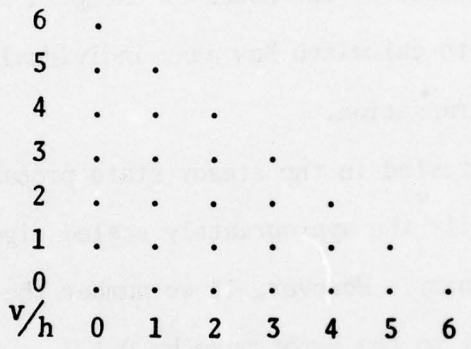
RANDØM(SEED)

This function subprogram returns a floating-point pseudo-random number between 0 and 1. It is used to initialize Q.

6. Numerical Examples

The program described above has been tested on a number of problems. In this section we give two examples that illustrate the flexibility of the method and its ability to deal with equimodular eigenvalues.

The first example is a random walk on an (n+1) × (n+1) triangular grid, which is illustrated below for n = 6.



The points of the grid are labelled (v,h) (v=0,...,n-h; h=0,...,n). From the point (v,h), a transition may take place to one of the four adjacent points (v+1,h), (v,h+1), (v-1,h), and (v,h-1). The probability of jumping to (v-1,h) or (v,h-1) is

$$pd(v,h) = (v+h)/n \tag{6.1}$$

with the probability being split equally between the two points when both are on the grid. The probability of jumping to (v+1,h) or (v,h+1) is

$$pu(v,h) = 1 - pd(v,h) \tag{6.2}$$

with the probability again being split when both points are on the grid.

If the $(n+1)(n+2)/2$ nodes (v,h) are numbered $1,2,\ldots,(n+1)(n+2)/2$ in some fashion, then the random walk can be expressed as a finite Markov chain whose transition matrix A consists of the probabilities a_{ij} of jumping from node j to node i (A is actually the transpose of the usual transition matrix; see [4]). To calculate the i-th element of the vector Aq one need only regard the components of q as the average number of individuals at the nodes of the grid and use the probabilities (6.1) and (6.2) to calculate how many individuals will be at node i after the next transition.

We are interested in the steady state probabilities of the chain, which is ordinarily the appropriately scaled eigenvector corresponding to the eigenvalue unity. However, if we number the diagonals on the grid that are parallel to the hypotenuse by $0,1,2,\ldots,n$, then an individual on an even diagonal can only jump to an odd diagonal, and vice versa. This means that the chain is cyclic with period two. Computationally it means that A has an eigenvalue of -1 as well as $+1$.

To run the problem on SRRIT, the nodes of the grid were matched with the components of the vector q in the order $(0,0),(1,0),\ldots,(n,0),(0,1),(1,1),\ldots,(n,1),(0,2),\ldots$. The subroutine that computes AQ is listed in the appendix. Note that the matrix A is never explicitly used; all computations are done in terms of the transition probabilities (6.1) and (6.2). The use of a common block to transmit information from the program that called SRRIT is typical.

The problem was run for a 30 × 30 grid which means N = 496. We took M = 6, NV = 4, and EPS = 10^{-5}. The results for each iteration in which an SRR step was performed are summarized in the following table. The variables ER and EI are the real and imaginary parts of the eigenvalues and RSD is the norm of the corresponding residual. CTR is the center of the current convergence cluster, AE is the average value of the eigenvalues in the cluster, and ARSD is the RMS average of the residuals. DSRR is the number of iterations to the next SRR step and DØRT is the number to the next orthogonalization.

IT = 0

ER	.9457+00	-.9096-01	-.4841-01	.3469-01	.3469-01	-.1921-01
EI	0	0	0	.1617-01	-.1617-01	0
RSD	.38+00	.60+00	.61+00	.59+00	.59+00	.63+00
CTR	.9457+00					
AE	.9457+00		DSRR=5	DØRT=1		
ARSD	.38+00					

IT = 5

ER	.1012+01	-.3912+00	.2400+00	-.1800+00	.1371+00	.3517-01
EI	0	0	0	0	0	0
RSD	.19+00	.84+00	.93+00	.89+00	.91+00	.93+00
CTR	.1012+01					
AE	.1012+01		DSRR=5	DØRT=1		
ARSD	.19+00					

IT = 10

ER	.1017+01	-.5987+00	-.3499+00	.3251+00	.9255-01	.5706-01
EI	0	0	0	0	0	0
RSD	.12+00	.75+00	.89+00	.92+00	.95+00	.95+00
CTR	.1017+01					
AE	.1017+01		DSRR=10	DØRT=1		
ARSD	.12+00					

IT = 20

ER	.1009+01	-.8751+00	.5175+00	-.5124+00	.3747+00	-.1485+00
EI	0	0	0	0	0	0
RSD	.58=01	.46+00	.82+00	.84+00	.88+00	.94+00
CTR	.1009+01					
AE	.1009+01		DSSR=20	DØRT=2		
ARSD	.58-01					

IT = 40

ER	.1001+01	-.9843+00	.9195+00	-.9144+00	.7946+00	-.5166+00
EI	0	0	0	0	0	0
RSD	.23-01	.14+00	.37+00	.40+00	.55+00	.95+00
CTR	.1001+01					
AE	.1001+01		DSSR=40	DØRT=1		
ARSD	.23-01					

IT = 80

ER	.1000+01	-.9998+00	.9935+00	-.9934+00	.8734+00	.2408+00
EI	0	0	0	0	0	0
RSD	.74-02	.29-01	.36-01	.78-02	.43+00	.92+00
CTR	.1000+01					
AE	.1991-93		DSRR=80	DØRT=3		
ARSD	.21-01					

IT = 160

ER	-.1000+01	.1000+01	.9935+00	-.9935+00	.9470+00	-.2138+00
EI	0	0	0	0	0	0
RSD	.56-03	.13-02	.38-03	.70-03	.23-03	.94-03
CTR	.1000+01					
AE	-.1304-04		DSRR=135	DØRT=2		
ARSD	.10-02					

IT = 295

ER	-.1000+01	.1000+01	.9935+00	-.9935+00	.9755+00	-.9738+00
EI	0	0	0	0	0	0
RSD	.30-04	.37-05	.13-05	.12-03	.84-02	.57-01
CTR	.1000+01		.9935+00			
AE	-.1863-06		.1080-06		DSRR=30	DØRT=30
ARSD	.21-04		.83-04			

IT = 325

ER	-.1000+01	.1000+01	.9935+00	-.9935+00	.9755+00	-.9751+00
EI	0	0	0	0	0	0
RSD	.70-05	.82-06	.35-06	.34-04	.39-02	.26-01
CTR	.1000+01		.9935+00			
AE	-.4470-07		.7451-07		DSRR=23	DØRT=23
ARSD	.50-05		.24-04			

IT = 348

ER			.9935+00	-.9935+00	.9755+00	-.9754+00
EI			0	0	0	0
RSD			.12-06	.12-04	.21-02	.15-01
CTR			.9935+00			
AE			.1118-06			
ARSD			.88-05			

The course of the iteration is unexceptionable. The program doubles the interval between SRR steps until it can predict convergence of the first cluster corresponding to the eigenvalues ±1. The first prediction falls slightly short, but the second gets it. After a third prediction the program terminates on the convergence of the second group of two eigenvalues.

It should be noted that the eigenvalue -1 has appeared as the dominant one. A transformation bringing the eigenvector corresponding to 1 can be obtained by calling EXCHNG of [11] to interchange the eigenvalues 1 and -1 (however, in this case the eigenvector corresponding to 1 is just the absolute value of the eigenvector corresponding to -1).

Without actually making timing runs, it is difficult to predict how much work is entailed in finding the eigenvalues. For example, runs were made with M = 2,4,6,8, which gave the following table of iterations required for the convergence of the first group of two eigenvalues.

m	it	m•it
2	1737	3474
4	523	2092
6	325	1950
8	188	1504

As predicted by the convergence theory, the number of iterations decreases as m increases. However, as m increases we must also multiply more columns of Q by A, and for this particular problem the number m•it is probably a better measure of the amount of work involved. From the table it is seen that this measure is also decreasing, although less dramatically than

the number of iterations. This of course does not include the overhead generated by SRRIT itself, which increases with m and may be considerable.

The second example shows how SRRIT can be used in conjunction with the inverse power method to find the smallest eigenvalues of a matrix. Consider the boundary value problem

$$
\begin{aligned}
& y'' + \mu^2 y = 0 , \\
& y(0) = 0 , \\
& y'(0) + \gamma y'(1) = 0, \quad 0 < \gamma < 1 .
\end{aligned}
\tag{6.3}
$$

The eigenvalues of this problem are easily seen to be given by

$$\mu = i \cosh^{-1} (-\gamma^{-1}) ,$$

which are complex. The following table lists the reciprocals of the first eight eigenvalues for $\gamma = 0.01$.

$$
\begin{array}{cc}
\mu^{-2} & |\mu^{-2}| \\
-0.01264 \pm 0.02313i & .02636 \\
0.004446 \pm 0.007308 & .008544 \\
0.002895 \pm 0.002204 & .003638 \\
0.001740 \pm 0.0008901 & .001954
\end{array}
\tag{6.4}
$$

The solution of (6.3) can be approximated by finite difference techniques as follows. Let y_i denote the approximate solution at the point $x_i = i/(n+1)$ $(i=0,1,\ldots,n+1)$. Replacing the derivatives in (6.3) with three point difference operators, we obtain the following generalized matrix eigenvalue problem for $y = (y_1, y_2, \ldots, y_{n+1})^T$:

$$Ay + \mu^2 By = 0 ,$$

where

$$A = \begin{bmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ 0 & 1 & -2 & 1 & & & \\ & & & \ddots & & & \\ & & & & 1 & -2 & 1 & 0 \\ & & & & 0 & 1 & -2 & 1 \\ 4 & 1 & 0 \ldots & 0 & \gamma & -4\gamma & 3\gamma & \end{bmatrix}$$

and $B = h^2$ diag $(1,1,\ldots,1,0)$. We may recast this problem in the form

$$Cy = \frac{1}{\mu^2} y ,$$

where $C = A^{-1}B$.

To apply SRRIT to this problem, we must be able to compute $z = Cq$ for any vector q. This can be done by solving the linear system

$$Az = Bq ,$$

which is easily done by sparse Gaussian elimination.

The problem was run for $n = 300$ with $M = 6$, $NV = 4$, and $EPS = 10^{-4}$. The results were the following:

IT = 0

ER	-.1525+00	.1179-02	.1548-03	.9887-04	.9887-04	.2577-04
EI	0	0	0	.5598-04	-.5598-04	0
RSD	.15+00	.85-02	.65-02	.15-01	.15-01	.71-02
CTR	.1525+00					
AE	-.1525+00		DSRR=5	DØRT=1		
ARSD	.15+00					

IT = 5

ER	-.1264-01	-.1264-01	.4438-02	.4438-02	.3104-02	.3104-02
EI	.2313-01	-.2313-01	.7323-02	-.7323-02	.2402-02	-.2402-02
RSD	.85-07	.85-07	.81-05	.81-05	.20-03	.20-03
CTR	.2636-01					
AE	.1264-01		DSRR=5	DØRT=1		
ARSD	.85-07					

IT = 10

ER	-.1264-01	-.1244-01	.4447-02	.4447-02	.2909-02	.2909-02
EI	.2313-01	-.2313-01	.7308-02	-.7308-02	.2204-02	-.2204-02
RSD	.60-08	.60-08	.16-07	.15-07	.93-05	.93-05
CTR	.2636-01		.8555-02			
AE	-.1264-01		.4447-02			
ARSD	.60-08		.15-07			

Given the extremely favorable ratios of the eigenvalues in Table (6.4) --the absolute value of the ratio of the seventh to the first is about .075 --it is not surprising that the iteration converges quickly. Indeed the only thing preventing convergence at the fifth iteration is that the first eigenvalue changed from real in the first iteration to complex in the fifth. Thus the problem is hardly a fair test of machinery of SRRIT. However, it is an excellent example how easy it is to apply SRRIT to a problem with complex eigenvalues. It also disposes of the notion that large eigenvalue problems must always require a large amount of work to solve; the factor that limits the size of n is the storage available, not the time required to compute Ax.

References

1. Bauer, F. L. Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischer Eigenwertprobleme, Z. Angew, Math. Phys. 8 (1957) 214-235.

2. Clint, M. and Jennings, A. The evaluation of eigenvalues and eigenvectors of a real symmetric matrix by simultaneous iteration, Comput. J. 13 (1970) 68-80.

3. _____. A simultaneous iteration method for the unsymmetric eigenvalue problem, J. Inst. Math. Appl. 8 (1971) 111-121.

4. Feller, W. An Introduction to Probability Theory and Its Applications, John Wiley, New York, 1961.

5. Rutishauser, H. Computational aspects of F. L. Bauer's simultaneous iteration method, Numer. Math. 13 (1969) 4-13.

6. _____. Simultaneous iteration method for symmetric matrices, Numer. Math. 16 (1970) 205-223 (also in [ ]).

7. Smith, B. T., Boyle, J. M., Garbow, B. S., Ikebe, Y., Klema, V.C., and Moler, C. B. Matrix Eigensystem Routines--EISPACK Guide, Lecture Notes in Computer Science, v. 6, Springer, New York, 1974.

8. Stewart, G. W. Accelerating the orthogonal iteration for the eigenvalues of a Hermitian matrix, Numer. Math. 13 (1969) 362-376.

9. _____. Introduction to Matrix Computations, Academic Press, New York, 1973.

10. _____. Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices, Numer. Math. 25 (1976) 123-136.

11. _____. HQR3 and EXCHNG: FORTRAN subroutines for calculating the eigenvalues of a real upper Hessenberg matrix in a prescribed order, ACM Trans. Math. Software 2 (1976) 275-280.

12. Wilkinson, J. H., and Reinsch, C. (eds.) Handbook for Automatic Computation, v. II. Linear Algebra, Springer, New York, 1971.

Algorithms

```
      SUBROUTINE SRRIT(Q,AQ,ATQ,N,NV,M,EPS,MAXIT,START,T,
     1                 ER,EI,TYPE,RSD,RSDX,WRITE)
C
C     PARAMETERS IN THE CALLING SEQUENCE
C
      INTEGER N,NV,M,MAXIT,START,TYPE(10),RSDX(10)
      REAL Q(500,10),AQ(500,10),EPS,T(10,10),ER(10),EI(10),RSD(10)
      LOGICAL WRITE
      EXTERNAL ATQ
C
C     SRRIT IS A FORTRAN SUBROUTINE TO COMPUTE A NESTED SEQUENCE
C     OF ORTHONORMAL BASES FOR THE DOMINANT INVARIANT SUBSPACES OF
C     A REAL MATRIX A OF ORDER N.  SPECIFICALLY, THE PROGRAM RETURNS
C     AN NXNV MATRIX Q WITH ORTHONORMAL COLUMNS AND AN NV X NV MATRIX T
C     SATISFYING
C
C               A*Q = Q*T + O(EPS).
C
C     THE MATRIX T IS QUASI-TRIANGULAR, THAT IS IT IS BLOCK
C     TRIANGULAR WITH 1X1 AND 2X2 BLOCKS ON ITS DIAGONAL.  THE
C     EIGENCALUES IN THE 1X1 BLOCKS ARE REAL.  THE
C     THE EIGENVALUES IN THE 2X2 BLOCKS ARE COMPLEX CONJUGATE
C     PAIRS.  THE EIGENVALUES E(1), E(2),..., E(N) ARE ORDERED
C     SO THAT
C
C           ABS(E(1)) .GE. ABS(E(2)) .GE.,...,.GE. ABS(E(NV)),
C
C     AND THESE EIGENVALUES APPROXIMATE THE LARGEST EIGENVALUES
C     OF A.  THESE FACTS HAVE THE FOLLOWING CONSEQUENCES.
C
C        1. IF E(L) .NE. E(L+1) AND E(L) .NE. CONJ(E(L+1)),
C           THEN COLUMNS 1,2,...,L OF Q FORM AN APPROXIMATE
C           BASIS FOR THE INVARIANT SUBSPACE CORRESPONDING TO
C           THE L LARGEST EIGENVALUES OF A.  THE LXL LEADING
C           PRINCIPAL SUBMATRIX OF T IS A REPRESENTATION OF
C           A IN THAT SUBSPACE WITH RESPECT TO THE BASIS Q.
C
C        2. IF Z IS AN EIGENVECTOR OF T CORRESPONDING TO E,
C           THEN Q*Z IS AN APPROXIMATE EIGENVECTOR OF A
C           CORRESPONDING TO E.
C
C     THE PROGRAM ACTUALLY ITERATES WITH AN NXM MATRIX Q
C     AND AN MXM MATRIX T.  SINCE THE RATE OF CONVERGENCE
C     OF THE L-TH COLUMN OF Q IS ESSENTIALLY LINEAR WITH
C     RATIO ABS(E(M+1)/E(L)), IT MAY PAY THE USER TO SET M
C     LARGER THAN THE NUMBER, NV, OF VECTORS HE WANTS TO
C     COMPUTE.
C
C     THE USER MUST FURNISH A SUBROUTINE TO COMPUTE THE
C     PRODUCT A*Q.  THE CALLING SEQUENCE IS
C
C             CALL ATQ(Q,AQ,L1,L2)
C
C     FOR J=L1,L1+1,...,L2 THE PROGRAM MUST PLACE THE PRODUCT
C     A*Q(*,J) IN AQ(*,J).
C
C     THE PARAMETERS IN THE CALLING SEQUENCE OF SRRIT ARE
C     (STARRED PARAMETERS ARE ALTERED BY THE PROGRAM)
C
C         *Q        AN ARRAY THAT ON RETURN CONTAINS THE
C                   ORTHONORMAL VECTORS DESCRIBED ABOVE.  INITIALLY
C                   Q MAY CONTAIN A STARTING APPROXIMATION
C                   (CF. START BELOW).
C         *AQ       AN ARRAY THAT ON RETURN CONTAINS THE PRODUCT
```

```
C                A*Q.
C          ATQ   THE NAME OF A SUBROUTINE TO EVALUATE THE
C                PRODUCT A*Q.
C          N     THE ORDER OF THE MATRIX A.
C         *NV    THE NUMBER OF VECTORS TO COMPUTE. ON RETURN,
C                NV CONTAINS THE NUMBER OF VECTORS THAT HAVE
C                CONVERGED.
C          M     THE NUMBER OF COLUMNS OF Q TO ITERATE WITH.
C          EPS   A CONVERGENCE CRITERION.
C          MAXIT AN UPPER BOUND ON THE NUMBER OF ITERATIONS
C                THE PROGRAM IS TO EXECUTE.
C          START AN INITIALIZING SIGNAL.  IF START .LT. 0,
C                Q IS INITIALIZED BY ORTHOGONALIZING A SET OF
C                RANDOM VECTORS.  IF START .GE. 0 THE COLUMNS
C                OF Q ARE USED AS A STARTING APPROXIMATION AND
C                IF START .GE. 1 THEY ARE ALSO ASSUMED TO BE
C                ORTHONORMAL.
C         *T     ON RETURN T CONTAINS THE REPRESENTATION OF A
C                DESCRIBED ABOVE.
C         *ER    AN ARRAY THAT ON RETURN CONTAINS THE REAL PARTS
C                OF THE EIGENVALUES OF T.
C         *EI    AN ARRAY THAT ON RETURN CONTAINS THE COMPLEX PARTS
C                OF THE EIGENVALUES OF T.
C         *TYPE  AN INTEGER ARRAY.  ON RETURN TYPE(L) CONTAINS
C                     0  IF THE L-TH EIGENVALUE IS REAL
C                     1  IF THE L-TH EIGENVALUE IS THE FIRST OF
C                        A COMPLEX CONJUGATE PAIR.
C                     2  IF THE L-TH EIGENVALUE IS THE SECOND OF
C                        A COMPLEX CONJUGATE PAIR.
C                    -1  IF THE L-TH EIGENVALUE WAS NOT CORRECTLY
C                        DETERMINED.
C
C         *RSD   AN ARRAY THAT ON RETURN CONTAINS THE 2-NORMS OF
C                THE RESIDUAL VECTORS  A*Q(*,L) - Q*T(*,L).
C         *RSDX  AN INTEGER ARRAY THAT ON RETURN CONTAINS
C                THE ITERATIONS AT WHICH THE RESIDUALS WERE COMPUTED.
C          WRITE A LOGICAL PARAMETER THAT, IF TRUE, CAUSES
C                INFORMATION ABOUT THE COURSE OF THE ITERATION TO BE
C                WRITED ON UNIT 6.
C
C
C     CONTROL PARAMETERS
C
      INTEGER INIT,SEED
      REAL ALPHA,BETA,CNVTOL,GRPTOL,ORTTOL,STPFAC
C
C     INTERNAL VARIABLES
C
      INTEGER DORT,DSRR,I,IT,J,L,NGRP,NOGRP,NXTORT,
     1        NXTSRR,ORSDX(10),OTYPE(10)
      REAL AE,AOE,AORSD,ARSD,CTR,OCTR,OEI(10),OER(10),ORSD(10)
C
C     INITIALIZE CONTROL PARAMETERS
C
      INIT = 5
      STPFAC = 2.
      SEED = 69
      ALPHA = 1.
      BETA  = 1.1
      GRPTOL = .001
      CNVTOL = .001
      ORTTOL = 2.
C
C     INITIALIZE
C
      L = 1
```

```
      IT = 0
      DO 10 J=1,M
        RSD(J) = 0.
        RSDX(J) = -1
        TYPE(J) = -1
   10 CONTINUE
      IF(START .GE. 0) GO TO 40
        DO 30 J=1,M
          DO 20 I=1,N
            Q(I,J) = RANDOM(SEED)
   20     CONTINUE
   30   CONTINUE
   40 CONTINUE
      IF(START .GT. 0) GO TO 50
        CALL ORTH(Q,1,M,N)
   50 CONTINUE
C
C     SRR LOOP
C
  100 CONTINUE
      IF (WRITE) WRITE(6,2000) IT,L
 2000 FORMAT(/10H SRR  IT =,I5,5H  L =,I3)
        CALL SRRSTP(Q,AQ,ATQ,L,M,N,T,ER,EI,TYPE,OER,OEI,OTYPE,
     1              WRITE)
        CALL RESID(Q,AQ,T,RSD,RSDX,ORSD,ORSDX,L,M,M,N,IT,TYPE,
     1             WRITE)
C
C       TEST FOR CONVERGENCE
C
  110   CONTINUE
          CALL GROUP(ER,EI,TYPE,RSD,GRPTOL,L,M,N,
     1               NGRP,CTR,AE,ARSD,WRITE)
          CALL GROUP(OER,OEI,OTYPE,ORSD,GRPTOL,L,M,N,
     1               NOGRP,OCTR,AOE,AORSD,WRITE)
          IF(NGRP .NE. NOGRP) GO TO 130
          IF(NGRP .EQ. 0) GO TO 130
          IF(ABS(AE-AOE) .GT. CTR*CNVTOL*FLOAT(RSDX(L)-ORSDX(L)))
     1                 GO TO 130
          IF(ARSD .GT. EPS) GO TO 130
          L = L + NGRP
          IF(L .GT. M) GO TO 130
        GO TO 110
  130   CONTINUE
        IF (WRITE) WRITE(6,2000) IT,L
C
C       EXIT IF THE REQUIRED NUMBER OF VECTORS HAVE CONVERGED.
C
        IF(L .GT. NV) GO TO 300
C
C       EXIT IF ITERATION COUNT EXCEEDS THE MAXIMUM NUMBER
C       OF ITERATIONS.
C
        IF(IT .GE. MAXIT) GO TO 300
C
C       DETERMINE WHEN THE NEXT SRR STEP IS TO BE TAKEN.
C
        NXTSRR = AMAX1(STPFAC*FLOAT(IT),FLOAT(INIT))
        NXTSRR = MINO(MAXIT,NXTSRR)
        DSRR = NXTSRR-IT
        IF(NGRP .NE. NOGRP) GO TO 150
        IF(NGRP .EQ. 0) GO TO 150
          IF(ARSD.GE.AORSD) GO TO 150
          DSRR = ALPHA + BETA*FLOAT(ORSDX(L)-RSDX(L))*ALOG(ARSD/EPS)/
     1                   ALOG(ARSD/AORSD)
          DSRR = MAX0(1,DSRR)
  150   CONTINUE
```

```
      NXTSRR = MINO(NXTSRR,IT+DSRR)
C
C     DETERMINE THE INTERVAL BETWEEN ORTHOGONALIZATIONS
C
      DORT = AMAX1(1.,ORTTOL/ALOG10(COND(T,M,WRITE)))
      NXTORT = MINO(IT+DORT,NXTSRR)
      IF (WRITE) WRITE(6,2001) IT,NXTSRR,NXTORT
 2001 FORMAT(/10H SRR  IT =,I5,10H  NXTSRR =,I5,10H  NXTORT =,I5)
      DO 157 J=L,M
        DO 153 I=1,N
          Q(I,J) = AQ(I,J)
  153   CONTINUE
  157 CONTINUE
      IT = IT+1
C
C     ORTHOGONALIZATION LOOP.
C
  160 CONTINUE
C
C       POWER LOOP
C
  170   CONTINUE
          IF(IT .EQ. NXTORT) GO TO 200
          CALL ATQ(Q,AQ,L,M)
          DO 190 J=L,M
            DO 180 I=1,N
              Q(I,J) = AQ(I,J)
  180       CONTINUE
  190     CONTINUE
          IT = IT + 1
          GO TO 170
  200     CONTINUE
          CALL ORTH(Q,L,M,N)
          NXTORT = MINO(IT+DORT,NXTSRR)
        IF(IT .LT. NXTSRR) GO TO 160
      GO TO 100
  300 CONTINUE
      NV = L-1
      RETURN
      END


      SUBROUTINE SRRSTP(Q,AQ,ATQ,L,M,N,T,ER,EI,TYPE,
     1                  OER,OEI,OTYPE,WRITE)
C
C     PARAMETERS IN THE CALLING SEQUENCE.
C
      INTEGER L,M,N,TYPE(10),OTYPE(10)
      REAL Q(500,10),AQ(500,10),T(10,10),ER(10),EI(10),
     1     OER(10),OEI(10)
      LOGICAL WRITE
      EXTERNAL ATQ
C
C     SRRSTP PERFORMS A SCHUR-RAYLEIGH-RITZ REFINEMENT ON
C     THE SET OF M ORTHONORMAL N-VECTORS CONTAINED IN
C     THE ARRAY G.  FIRST THE SUBROUTINE ATQ IS CALLED
C     TO GENERATE THE PRODUCT OF THE MATRIX A AND THE
C     VECTORS Q IN THE ARRAY Q.  THEN THE MATRIX
C     T=TR(Q)*AQ IS REDUCED TO ORDERED QUASI-TRIANGULAR
C     FORM BY THE SUBROUTINE OTHES, ORTRAN AND HQR3.
C     THE REDUCING TRANSFORMATION V IS POSTMULTIPLIED
C     INTO Q AND AQ TO GIVE THE REFINDED VECTORS IN Q AND
C     THEIR PRODUCT WITH A IN AQ.  IT IS ASSUMENLD THAT IT IS
C     NECESSARY TO WORK WITH ONLY COLUMNS L THROUGH M OF T.
C     THE INFORMATION CONTAINED IN POSITIONS L THROUGH M
C     OF THE ARRAYS ER, EI, AND TYPE IS STORED IN THE
```

```
C     CORRESPONDING POSITIONS OF THE ARRAYS OER, OEI AND OTYPE.
C
C     INTERNAL VARIABLES
C
      INTEGER I,J,K
      REAL AP(10),P(10),V(10,10),MCHEPS
C
      IF (WRITE) WRITE(6,2000) L
 2000 FORMAT(/12H SRRSTP  L =,I5)
      MCHEPS = 1.
    3 CONTINUE
        IF(MCHEPS+1. .EQ. 1.) GO TO 5
        MCHEPS = MCHEPS/2.
        GO TO 3
    5 CONTINUE
C
C     SAVE THE OLD EIGENVALUES.
C
      DO 10 J=L,M
        OER(J) = ER(J)
        OEI(J) = EI(J)
        OTYPE(J) = TYPE(J)
   10 CONTINUE
C
C     CALCULATE THE NEW T.
C
      CALL ATQ(Q,AQ,L,M)
      DO 40 J=L,M
        DO 30 I=1,M
          T(I,J) = 0.
          DO 20 K=1,N
            T(I,J) = T(I,J) + Q(K,I)*AQ(K,J)
   20     CONTINUE
   30   CONTINUE
   40 CONTINUE
C
C     TRIANGULARIZE T
C
C
      CALL ORTHES(10,M,L,M,T,P)
      CALL ORTRAN(10,M,L,M,T,P,V)
      CALL HQR3(T,V,M,L,M,MCHEPS,ER,EI,TYPE,10,10)
      IF (.NOT.WRITE) GO TO 48
        WRITE(6,1001)
        DO 43 I=1,M
          WRITE(6,1000) (T(I,J),J=1,M)
   43   CONTINUE
        WRITE(6,1002)
        DO 45 I=1,M
          WRITE(6,1000) (V(I,J),J=1,M)
   45   CONTINUE
        WRITE(6,1003)
        WRITE(6,1000) (ER(I),I=1,M)
        WRITE(6,1004)
        WRITE(6,1000) (EI(I),I=1,M)
        WRITE(6,1005)
        WRITE(6,1100) (TYPE(I),I=1,M)
 1000   FORMAT(/1H ,10E12.4)
 1001   FORMAT(//2H T)
 1002   FORMAT(//2H V)
 1003   FORMAT(//3H ER)
 1004   FORMAT(3H EI)
 1005   FORMAT(//5H TYPE)
 1100   FORMAT(/1H ,10I12)
C
   48 CONTINUE
```

```
C
C     TRANSFORM Q AND AQ
C
      DO 80 I=1,N
        DO 60 J=L,M
          P(J) = 0.
          AP(J) = 0.
          DO 50 K=1,M
            P(J) = P(J) + Q(I,K)*V(K,J)
            AP(J) = AP(J) + AQ(I,K)*V(K,J)
   50     CONTINUE
   60   CONTINUE
        DO 70 J=L,M
          Q(I,J) = P(J)
          AQ(I,J) = AP(J)
   70   CONTINUE
   80 CONTINUE
      RETURN
      END


      SUBROUTINE GROUP(ER,EI,TYPE,RSD,GRPTOL,L,M,N,NGRP,CTR,AE,ARSD,
     1                 WRITE)
C
C     PARAMETERS IN THE CALLING SEQUENCE.
C
      INTEGER TYPE(10),L,M,N,NGRP
      REAL ER(10),EI(10),RSD(10),GRPTOL,CTR,AE,ARSD
      LOGICAL WRITE
C
C     GROUP IS A SUBROUTINE TO FIND A CLUSTER OF COMPLEX
C     NUMBERS WHOSE REAL PARTS ARE CONTAINED IN THE ARRAY
C     ER AND IMAGINARY PARTS ARE CONTAINED IN THE ARRAY EI.
C     THESE NUMBERS ARE ASSUMED TO BE STORED IN DESCENDING
C     ORDER OF MAGNITUDE.  NGRP IS DETERMINED AS THE LARGEST
C     INTEGER LESS THAN OR EQUAL TO M FOR WHICH THE ABSOLUTE
C     VALUE E(J) OF THE NUMBER ER(J)+EI(J)*I SATISFIES
C
C                  E(L) - E(L+NGRP-1) <= GRPTOL / 2.
C
C     AND FOR WHICH TYPE(L),TYPE(L+1),...,TYPE(L+NGRP-1) IS
C     NONNEGATIVE.  IF NGRP=0, THE SUBROUTINE RETURNS
C     CTR=AE=ARSD=0.  IF NGRP.NE.0, CTR IS SET TO
C     (E(L)+E(L+NGRP-1))/2, AE TO THE AVERAGE OF THE
C     NUMBERS ER+EI*I, AND ARSD TO THE RMS AVERAGE OF
C     RSD(L),RSD(L+1),...,RSD(L+NGRP-1).
C
C
C     INTERNAL VARIABLES.
C
      INTEGER J,L1
      REAL MOD,MOD1
      NGRP = 0
      MOD = SQRT(ER(L)**2 + EI(L)**2)
      CTR = 0.
   10 CONTINUE
        L1 = L + NGRP
        IF(L1.GT.M .OR. TYPE(L1).LT.0) GO TO 20
        MOD1 = SQRT(ER(L1)**2 + EI(L1)**2)
        IF(ABS(MOD-MOD1) .GT. GRPTOL*(MOD+MOD1)) GO TO 20
        CTR = (MOD + MOD1)/2.
        NGRP = NGRP + TYPE(L1) + 1
        GO TO 10
   20 CONTINUE
      AE = 0.
      ARSD = 0.
```

```
      IF(NGRP .EQ. 0) GO TO 40
        L1 = L+NGRP-1
        DO 30 J=L,L1
          AE = AE + ER(J)
          ARSD = ARSD + RSD(J)**2
   30   CONTINUE
        AE = AE/FLOAT(NGRP)
        ARSD = SQRT(ARSD/FLOAT(NGRP))
   40 CONTINUE
      WRITE(6,2000) NGRP,CTR,AE,ARSD
 2000 FORMAT(/14H GROUP  NGRP =,I5,7H  CTR =,E12.4,6H  AE =,E12.4,
     1       8H  ARSD =,E12.4)
      RETURN
      END


      SUBROUTINE RESID(Q,AQ,T,RSD,RSDX,ORSD,ORSDX,L1,L2,M,N,IT,TYPE,
     1                 WRITE)
C
C     PARAMETERS IN THE CALLING SEQUENCE.
C
      INTEGER RSDX(10),ORSDX(10),L1,L2,M,N,IT,TYPE(10)
      REAL Q(500,10),AQ(500,10),T(10,10),RSD(10),ORSD(10)
      LOGICAL WRITE
C
C     RESID COMPUTES RESIDUALS CORRESPONDING TO EIGENVALUES
C     L1 THROUGH L2 OF THE QUASI-TRIANGULAR MATRIX T OF
C     ORDER M.  SPECIFICALLY IF T(*,J) IS THE J-TH COLUMN
C     OF T, RSD(J) IS SET TO NORM(AQ(*,J)-QT(*,J)), WHERE THE
C     NORM IS THE EUCLIDEAN NORM.  THE INDEX RSDX(J) IS SET
C     EQUAL TO IT.  IF THE J-TH EIGENVALUE IS COMPLEX
C     (TYPE(J) = 1) THE RMS AVERAGE OF RSD(J) AND RSD(J+1)
C     IS PLACED IN RSD(J) AND RSD(J+1).  THE INITIAL VALUES OF
C     RSD AND RSDX ARE STORED IN ORSD AND ORSDX.
C
C
C     INTERNAL VARIABLES
C
      INTEGER I,J,K,KU
      REAL S
C
      IF (WRITE) WRITE(6,2000) L1,L2
 2000 FORMAT(/12H RESID  L1 =,I5,6H  L2 =,I5)
      IF(L1 .GT. L2) RETURN
      DO 30 J=L1,L2
        ORSD(J) = RSD(J)
        ORSDX(J) = RSDX(J)
        RSDX(J) = IT
        KU = MINO(J+1,M)
        IF(TYPE(J) .EQ. 0) KU = J
        RSD(J) = 0.
        DO 20 I=1,N
          S = 0.
          DO 10 K=1,KU
            S = S + Q(I,K)*T(K,J)
   10     CONTINUE
          RSD(J) = RSD(J) + (AQ(I,J)-S)**2
   20   CONTINUE
   30 CONTINUE
      DO 50 J=L1,L2
        IF(TYPE(J) .NE. 1) GO TO 40
          RSD(J) = (RSD(J) + RSD(J+1))/2.
          RSD(J+1) = RSD(J)
   40   CONTINUE
        RSD(J) = SQRT(RSD(J))
   50 CONTINUE
```

```
      IF (.NOT. WRITE) GO TO 60
        WRITE(6,1000)
        WRITE(6,1001) (RSD(I),I=1,M)
        WRITE(6,1002)
        WRITE(6,1003) (RSDX(I),I=1,M)
 1000   FORMAT(//4H RSD)
 1001   FORMAT(/1H ,10E12.4)
 1002   FORMAT(//5H RSDX)
 1003   FORMAT(/1H ,10I12)
C
   60 CONTINUE
      RETURN
      END


      SUBROUTINE ORTH(Q,L,M,N)
C
C     PARAMETERS IN THE CALLING SEQUENCE
C
      INTEGER L,M,N
      REAL Q(500,10)
C
C     ORTH ORTHONORMALIZES COLUMNS L THROUGH M OF THE ARRAY
C     Q WITH RESPECT TO COLUMN 1 THROUGH M.  COLUMNS 1
C     THROUGH L-1 ARE ASSUMED TO BE ORTHONORMAL.  THE METHOD
C     IS THE GRAM-SCHMIDT METHOD WITH REORTHOGONCELIZATION.
C     A COLUMN IS ACCEPTED WHEN AN ORTHOGONALIZATION DOES
C     NOT REDUCE ITS EUCLIDEAN NORM BY A FACTOR OF MORE
C     THAN TOL.  IF THIS IS NOT DONE IN MAXTRY ATTEMPTS
C     THE PROGRAM STOPS.  THE PROGRAM ALSO STOPS IF IT
C     ENCOUNTERS A ZERO VECTOR.
C
C     INTERNAL CONTROL PARAMETERS
C
      REAL TOL
      INTEGER MAXTRY
C
C     INTERNAL VARIABLES
C
      REAL NORM,QQ
      INTEGER I,J,JM1,K,TRY
      MAXTRY = 5
      TOL = .5
      DO 160 J=L,M
C
C       ORTHOGONALIZE THE J-TH VECTOR
C
        JM1 = J-1
        TRY = 0
  100   CONTINUE
C
C         COMPUTE THE NORM OF THE VECTOR.
C
          NORM = 0.
          DO 110 I=1,N
            NORM = NORM + Q(I,J)**2
  110     CONTINUE
          NORM = SQRT(NORM)
C
C         ERROR TEST
C
          IF(NORM .EQ. 0.) GO TO 170
C
C         SCALE THE VECTOR.
C
          DO 120 I=1,N
```

```
            Q(I,J) = Q(I,J)/NORM
  120     CONTINUE
C
C         TEST TO SEE IF THE J-TH VECTOR IS ORTHOGONAL.
C
          IF(J .EQ. 1) GO TO 160
          IF(TRY .EQ. 0) NORM = 0.
          IF(NORM .GT. TOL) GO TO 160
          TRY = TRY + 1
          IF(TRY .GT. MAXTRY) GO TO 170
C
C         PERFORM ONE MODIFIED GRAM-SCHMIDT STEP.
C
          DO 150 K=1,JM1
            QQ = 0.
            DO 130 I=1,N
              QQ = QQ + Q(I,K)*Q(I,J)
  130       CONTINUE
            DO 140 I=1,N
              Q(I,J) = Q(I,J) - QQ*Q(I,K)
  140       CONTINUE
  150     CONTINUE
        GO TO 100
  160 CONTINUE
      RETURN
C
  170 CONTINUE
      WRITE(6,2000)
 2000 FORMAT(/14H ERROR IN ORTH)
      STOP
      END


      REAL FUNCTION COND(T,M,WRITE)
C
C     PARAMETERS IN THE CALLING SEQUENCE
C
      REAL T(10,10)
      INTEGER M
      LOGICAL WRITE
C
C     COND IS A FUNCTION THAT RETURNS THE CONDITION
C     NUMBER WITH RESPECT TO THE ROW-SUM NORM OF THE UPPER
C     HESSENBERG MATRIX T OF ORDER M.
C
C
C     INTERNAL VARIABLES
C
      REAL MULT(10),NT,NTR,NT1,NT1R,T1(10,10)
      INTEGER I,I1,J,JM1,J1,K,PVT(10)
      MM1 = M-1
      NT = 0.
      DO 20 I=1,M
        I1 = MAX0(I-1,1)
        NTR = 0.
        DO 10 J=I1,M
          T1(I,J) = T(I,J)
          NTR = NTR + ABS(T(I,J))
   10   CONTINUE
        NT = AMAX1(NT,NTR)
   20 CONTINUE
      DO 60 I=1,MM1
        PVT(I) = 0
        MULT(I) = 0.
        IF(T1(I+1,I) .EQ. 0.) GO TO 60
        IF(ABS(T1(I+1,I)).LE. ABS(T1(I,I))) GO TO 40
```

```
            PVT(I) = 1
            DO 30 J=I,M
              S = T1(I,J)
              T1(I,J) = T1(I+1,J)
              T1(I+1,J) = S
 30         CONTINUE
 40       CONTINUE
          MULT(I) = T1(I+1,I)/T1(I,I)
          T1(I+1,I) = 0.
          I1 = I+1
          DO 50 J=I1,M
            T1(I+1,J) = T1(I+1,J) - MULT(I)*T1(I,J)
 50       CONTINUE
 60     CONTINUE
        DO 110 J=1,M
          IF(T1(J,J) .NE. 0.) GO TO 70
            COND = 1.E8
            RETURN
 70       CONTINUE
           T1(J,J) = 1./T1(J,J)
           IF(J .EQ. 1) GO TO 100
             JM1 = J-1
             DO 90 I=1,JM1
               S = 0.
               DO 80 K=I,JM1
                 S = S + T1(I,K)*T1(K,J)
 80            CONTINUE
               T1(I,J) = -S*T1(J,J)
 90          CONTINUE
100       CONTINUE
110     CONTINUE
        DO 160 JJ=1,MM1
          J = M-JJ
          J1 = J+1
          IF(MULT(J) .EQ. 0.) GO TO 130
            DO 120 I=1,J1
              T1(I,J) = T1(I,J) - MULT(J)*T1(I,J+1)
120         CONTINUE
130       CONTINUE
          IF(PVT(J) .EQ. 0) GO TO 150
          DO 140 I=1,J1
              S = T1(I,J)
              T1(I,J) = T1(I,J+1)
              T1(I,J+1) = S
140       CONTINUE
150       CONTINUE
160     CONTINUE
        NT1 = 0.
        DO 180 I=1,M
          IM1 = MAX0(1,I-1)
          NT1R = 0.
          DO 170 J=IM1,M
            NT1R = NT1R + ABS(T1(I,J))
170       CONTINUE
          NT1 = AMAX1(NT1,NT1R)
180     CONTINUE
        COND = NT*NT1
        IF (WRITE) WRITE(6,2000) NT,NT1,COND
2000    FORMAT(/11H COND  NT =,E12.4,6H  NT1=,E12.4,8H  COND =,E12.4)
        RETURN
        END


        FUNCTION RANDOM(SEED)
        INTEGER SEED
C       RANDOM IS A FUNCTION THAT PRODUCES A PSEUDO-RANDOM
```

```
C     FLOATING POINT NUMBER IN THE INTERVAL FROM ZERO TO ONE.
      SEED = MOD(4621*SEED+2113,10000)
      RANDOM = FLOAT(SEED)/1.E4
      RETURN
      END


      SUBROUTINE HQR3(A,V,N,NLOW,NUP,EPS,ER,EI,TYPE,NA,NV)
C
      INTEGER N,NA,NLOW,NUP,NV,TYPE(N)
      REAL A(NA,N),EI(N),ER(N),EPS,V(NV,N)
C
C     HQR3 REDUCES THE UPPER HESSENBERG MATRIX A TO QUASI-
C     TRIANGULAR FORM BY UNITARY SIMILARITY TRANSFORMATIONS.
C     THE EIGENVALUES OF A, WHICH ARE CONTAINED IN THE 1X1
C     AND 2X2 DIAGONAL BLOCKS OF THE REDUCED MATRIX, ARE
C     ORDERED IN DESCENDING ORDER OF MAGNITUDE ALONG THE
C     DIAGONAL.  THE TRANSFORMATIONS ARE ACCUMULATED IN THE
C     ARRAY V.  HQR3 REQUIRES THE SUBROUTINES EXCHNG,
C     QRSTEP, AND SPLIT.  THE PARAMETERS IN THE CALLING
C     SEQUENCE ARE (STARRED PARAMETERS ARE ALTERED BY THE
C     SUBROUTINE)
C
C        *A       AN ARRAY THAT INITIALLY CONTAINS THE N X N
C                 UPPER HESSENBERG MATRIX TO BE REDUCED.  ON
C                 RETURN A CONTAINS THE REDUCED, QUASI-
C                 TRIANGULAR MATRIX.
C        *V       AN ARRAY THAT CONTAINS A MATRIX INTO WHICH
C                 THE REDUCING TRANSFORMATIONS ARE TO BE
C                 MULTIPLIED.
C         N       THE ORDER OF THE MATRICES A AND V.
C         NLOW    A(NLOW-1,NLOW) AND A(NUP,NUP+U) ARE
C         NUP     ASSUMED TO BE ZERO, AND ONLY ROWS NLOW
C                 THROUGH NUP AND COLUMNS NLOW THROUGH
C                 NUP ARE TRANSFORMED, RESULTING IN THE
C                 CALCULATION OF EIGENVALUES NLOW
C                 THROUGH NUP.
C         EPS     A CONVERGENCE CRITERION.
C        *ER      AN ARRAY THAT ON RETURN CONTAINS THE REAL
C                 PARTS OF THE EIGENVALUES.
C        *EI      AN ARRAY THAT ON RETURN CONTAINS THE
C                 IMAGINARY PARTS OF THE EIGENVALUES.
C        *TYPE    AND INTEGER ARRAY WHOSE I-TH ENTRY IS
C                   0   IF THE I-TH EIGENVALUE IS REAL,
C                   1   IF THE I-TH EIGENVALUE IS COMPLEX
C                       WITH POSITIVE IMAGINARY PART.
C                   2   IF THE I-TH EIGENVALUE IS COMPLEX
C                       WITH NEGATIVE IMAGINARY PART,
C                  -1   IF THE I-TH EIGENVALUE WAS NOT
C                       CALCULATED SUCCESSFULLY.
C         NA      THE FIRST DIMENSION OF THE ARRAY A.
C         NV      THE FIRST DIMENSION OF THE ARRAY V.
C
C     INTERNAL VARIABLES
C
      INTEGER I,IT,L,MU,NL,NU
      REAL E1,E2,P,Q,R,S,T,W,X,Y,Z
      LOGICAL FAIL
C
C     INITIALIZE.
C
      DO 10 I=NLOW,NUP
        TYPE(I) = -1
   10 CONTINUE
      T = 0.
C
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC


```
C     MAIN LOOP. FIND AND ORDER EIGENVALUES.
C
      NU = NUP
  100 IF(NU .LT. NLOW) GO TO 500
        IT = 0
C
C       QR LOOP.  FIND NEGLIGABLE ELEMENTS AND PERFORM
C       QR STEPS.
C
  110   CONTINUE
C
C         SEARCH BACK FOR NEGLIGABLE ELEMENTS.
C
          L = NU
  120     CONTINUE
            IF(L .EQ. NLOW) GO TO 130
            IF(ABS(A(L,L-1)) .LT. EPS*(ABS(A(L-1,L-1))+ABS(A(L,L))))
     1        GO TO  130
            L = L-1
          GO TO 120
  130     CONTINUE
C
C         TEST TO SEE IF AN EIGENVALUE OR A 2X2 BLOCK
C         HAS BEEN FOUND.
C
          X = A(NU,NU)
          IF(L .EQ. NU) GO TO 300
          Y = A(NU-1,NU-1)
          W = A(NU,NU-1)*A(NU-1,NU)
          IF(L  .EQ. NU-1) GO TO 200
C
C         TEST ITERATION COUNT. IF IT IS 30 QUIT.  IF
C         IT IS 10 OR 20 SET UP AN AD-HOC SHIFT.
C
          IF(IT .EQ. 30) GO TO 500
          IF(IT.NE.10 .AND. IT.NE.20) GO TO 150
C
C           AD-HOC SHIFT.
C
            T = T + X
            DO 140 I=NLOW,NU
              A(I,I) = A(I,I) - X
  140       CONTINUE
            S = ABS(A(NU,NU-1)) + ABS(A(NU-1,NU-2))
            X = 0.75*S
            Y = X
            W = -0.4375*S**2
  150     CONTINUE
          IT = IT + 1
C
C         LOOK FOR TWO CONSECUTIVE SMALL SUB-DIAGONAL
C         ELEMENTS.
C
          NL= NU-2
  160     CONTINUE
            Z =  A(NL,NL)
            R = X - Z
            S = Y - Z
            P = (R*S-W)/A(NL+1,NL) + A(NL,NL+1)
            Q = A(NL+1,NL+1) - Z - R - S
            R = A(NL+2,NL+1)
            S = ABS(P) + ABS(Q) + ABS(R)
            P = P/S
            Q = Q/S
            R = R/S
            IF(NL .EQ. L) GO TO 170
```

```
            IF(ABS(A(NL,NL-1))*(ABS(Q)+ABS(R)) .LE.
     1         EPS*ABS(P)*(ABS(A(NL-1,NL-1))+ABS(Z)+ABS(A(NL+1,NL+1))))
     2         GO TO 170
              NL = NL-1
            GO TO 160
  170       CONTINUE
C
C           PERFORM A QR STEP BETWEEN NL AND NU.
C
            CALL QRSTEP(A,V,P,Q,R,NL,NU,N,NA,NV)
          GO TO 110
C
C         2X2 BLOCK FOUND.
C
  200     IF(NU .NE. NLOW+1) A(NU-1,NU-2) = 0.
          A(NU,NU) = A(NU,NU) + T
          A(NU-1,NU-1) = A(NU-1,NU-1) + T
          TYPE(NU) = 0
          TYPE(NU-1) = 0
          MU = NU
C
C         LOOP TO POSITION  2X2 BLOCK.
C
  210     CONTINUE
            NL = MU-1
C
C           ATTEMPT  TO SPLIT THE BLOCK INTO TWO REAL
C           EIGENVALUES.
C
            CALL SPLIT(A,V,N,NL,E1,E2,NA,NV)
C
C           IF THE SPLIT WAS SUCCESSFUL, GO AND ORDER THE
C           REAL EIGENVALUES.
C
            IF(A(MU,MU-1) .EQ. 0.) GO TO 310
C
C           TEST TO SEE IF THE BLOCK IS PROPERLY POSITIONED,
C           AND IF NOT EXCHANGE IT
C
            IF(MU .EQ. NUP) GO TO 400
            IF(MU .EQ. NUP-1) GO TO 220
            IF(A(MU+2,MU+1) .EQ. 0.) GO TO 220
C
C             THE NEXT BLOCK IS 2X2.
C
              IF(A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1)
     1           .GE. A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*
     2                A(MU+2,MU+1))
     3           GO TO 400
              CALL EXCHNG(A,V,N,NL,2,2,EPS,FAIL,NA,NV)
              IF(.NOT. FAIL) GO TO 215
                TYPE(NL) = -1
                TYPE(NL+1) = -1
                TYPE(NL+2) = -1
                TYPE(NL+3) = -1
                GO TO 500
  215         CONTINUE
              MU = MU+2
              GO TO 230
  220       CONTINUE
C
C             THE NEXT BLOCK IS 1X1.
C
              IF(A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1)
     1           .GE. A(MU+1,MU+1)**2)
     2           GO TO 400
```

```
            CALL EXCHNG(A,V,N,NL,2,1,EPS,FAIL,NA,NV)
            IF(.NOT. FAIL) GO TO 225
              TYPE(NL) = -1
              TYPE(NL+1) = -1
              TYPE(NL+2) = -1
              GO TO 500
  225       CONTINUE
            MU = MU+1
  230     CONTINUE
        GO TO 210
C
C       SINGLE EIGENVALUE FOUND.
C
  300   NL = 0
        A(NU,NU) = A(NU,NU) + T
        IF(NU .NE. NLOW) A(NU,NU-1) = 0.
        TYPE(NU) = 0
        MU = NU
C
C       LOOP TO POSITION ONE OR TWO REAL EIGENVALUES.
C
  310   CONTINUE
C
C         POSITION THE EIGENVALUE LOCATED AT A(NL,NL).
C
  320     CONTINUE
            IF(MU .EQ. NUP) GO TO 350
            IF(MU .EQ. NUP-1) GO TO 330
            IF(A(MU+2,MU+1) .EQ. 0.) GO TO 330
C
C             THE NEXT BLOCK IS 2X2.
C
              IF(A(MU,MU)**2 .GE.
     1           A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*A(MU+2,MU+1))
     2          GO TO 400
              CALL EXCHNG(A,V,N,MU,1,2,EPS,FAIL,NA,NV)
            IF(.NOT. FAIL) GO TO 325
              TYPE(MU) = -1
              TYPE(MU+1) = -1
              TYPE(MU+2) = -1
              GO TO 500
  325         CONTINUE
              MU = MU+2
              GO TO 340
  330         CONTINUE
C
C             THE NEXT BLOCK IS 1X1.
C
              IF(ABS(A(MU,MU)) .GE. ABS(A(MU+1,MU+1)))
     1          GO TO 350
              CALL EXCHNG(A,V,N,MU,1,1,EPS,FAIL,NA,NV)
              MU = MU+1
  340         CONTINUE
          GO TO 320
  350     CONTINUE
          MU = NL
          NL = 0
        IF(MU .NE. 0) GO TO 310
C
C       GO BACK AND GET THE NEXT EIGENVALUE.
C
  400   CONTINUE
        NU = L-1
      GO TO 100
C
C     ALL THE EIGNVALUES HAVE BEEN FOUND AND ORDERED.
```

```
C     COMPUTE THEIR VALUES AND TYPE.
C
  500 IF(NU .LT. NLOW) GO TO 507
        DO 503 I=1,NU
          A(I,I) = A(I,I) + T
  503   CONTINUE
  507 CONTINUE
      NU = NUP
  510 CONTINUE
        IF(TYPE(NU) .NE. -1) GO TO 515
          NU = NU-1
          GO TO 540
  515   CONTINUE
        IF(NU .EQ. NLOW) GO TO 520
        IF(A(NU,NU-1) .EQ. 0.) GO TO 520
C
C        2X2 BLOCK.
C
          CALL SPLIT(A,V,N,NU-1,E1,E2,NA,NV)
          IF(A(NU,NU-1) .EQ. 0.) GO TO 520
          ER(NU) = E1
          EI(NU-1) = E2
          ER(NU-1) = ER(NU)
          EI(NU) = -EI(NU-1)
          TYPE(NU-1) = 1
          TYPE(NU) = 2
          NU = NU-2
          GO TO 530
  520   CONTINUE
C
C         SINGLE ROOT.
C
          ER(NU) = A(NU,NU)
          EI(NU) = 0.
          NU = NU-1
  530   CONTINUE
  540 CONTINUE
      IF(NU .GE. NLOW) GO TO 510
      RETURN
      END


      SUBROUTINE EXCHNG(A,V,N,L,B1,B2,EPS,FAIL,NA,NV)
C
      INTEGER B1,B2,L,NA,NV
      REAL A(NA,N),EPS,V(NV,N)
      LOGICAL FAIL
C
C     GIVEN THE UPPER HESSENBERG MATRIX A WITH CONSECUTIVE
C     B1XB1 AND B2XB2 DIAGONAL BLOCKS (B ,B2 .3E. 2)
C     STARTING AT A(L,L), EXCHNG PRODUCES A UNITARY
C     SIMILARITY TRANSFORMATION THAT EXCH+5GES THE B36@2S
C     ALONG WITH THEIR EIGENVALUES.  THE TRANSFORMATION
C     IS ACCUMULATED IN V.  EXCHNG REQUIRES THE SUBROUTINE
C     ORSTEP.  THE PARAMETERS IN THE CALLING SEQUENCE ARE
C     (STARRED PARAMETERS ARE ALTERED BY THE SUBROUTINE)
C
C        *A       THE MATRIX WHOSE BLOCKS ARE TO BE
C                 INTERCHANGED.
C        *V       THE ARRAY INTO WHICH THE TRANSFORMATIONS
C                 ARE TO BE ACCUMULATED.
C         N       THE ORDER OF THE MATRIX A.
C         L       THE POSITION OF THE BLOCKS.
C         B1      THE SIZE OF THE FIRST BLOCK.
C         B2      THE SIZE OF THE SECOND BLOCK.
C         EPS     A CONVERGENCE CRITERION.
```

```
C          *FAIL    A LOGICAL VARIABLE WHICH IS FALSE ON A
C                   NORMAL RETURN.  IF THIRTY ITERATIONS WERE
C                   PERFORMED WITHOUT CONVERGENCE, FAIL IS SET
C                   TO TRUE AND THE ELEMENT
C                   A(L+B2,L+B2-1) CANNOT BE ASSUMED ZERO.
C           NA      THE FIRST DIMENSION OF THE ARRAY A.
C           NV      THE FIRST DIMENSION OF THE ARRAY V.
C
C       INTERNAL VARIABLES.
C
        INTEGER I,IT,J,L1,M
        REAL P,Q,R,S,W,X,Y,Z
C
        FAIL = .FALSE.
        IF(B1 .EQ. 2) GO TO 40
          IF(B2 .EQ. 2) GO TO 10
C
C           INTERCHANGE 1X1 AND 1X1 BLOCKS.
C
            L1 = L+1
            Q = A(L+1,L+1) - A(L,L)
            P = A(L,L+1)
            R = AMAX1(P,Q)
            IF(R .EQ. 0.) RETURN
            P = P/R
            Q = Q/R
            R = SQRT(P**2 + Q**2)
            P = P/R
            Q = Q/R
            DO 3 J=L,N
              S = P*A(L,J) + Q*A(L+1,J)
              A(L+1,J) = P*A(L+1,J) - Q*A(L,J)
              A(L,J) = S
    3       CONTINUE
            DO 5 I=1,L1
              S = P*A(I,L) + Q*A(I,L+1)
              A(I,L+1) = P*A(I,L+1) - Q*A(I,L)
              A(I,L) = S
    5       CONTINUE
            DO 7 I=1,N
              S = P*V(I,L) + Q*V(I,L+1)
              V(I,L+1) = P*V(I,L+1) - Q*V(I,L)
              V(I,L) = S
    7       CONTINUE
            A(L+1,L) = 0.
            RETURN
   10     CONTINUE
C
C           INTERCHANGE 1X1 AND 2X2 BLOCKS.
C
            X = A(L,L)
            P = 1.
            Q = 1.
            R = 1.
            CALL QRSTEP(A,V,P,Q,R,L,L+2,N,NA,NV)
            IT = 0
   20       IT = IT+1
              IF(IT .LE. 30) GO TO 30
                FAIL = .TRUE.
                RETURN
   30         CONTINUE
              P = A(L,L) - X
              Q = A(L+1,L)
              R = 0.
              CALL QRSTEP(A,V,P,Q,R,L,L+2,N,NA,NV)
              IF(ABS(A(L+2,L+1)) .GT.
```

```
     1                              EPS*(ABS(A(L+1,L+1))+ABS(A(L+2,L+2))))
     1         GO TO 20
           A(L+2,L+1) = 0.
           RETURN
    40 CONTINUE
C
C        INTERCHANGE 2X2 AND B2XB2 BLOCKS.
C
           M = L+2
           IF(B2 .EQ. 2) M = M+1
           X = A(L+1,L+1)
           Y = A(L,L)
           W = A(L+1,L)*A(L,L+1)
           P = 1.
           Q = 1.
           R = 1.
           CALL QRSTEP(A,V,P,Q,R,L,M,N,NA,NV)
           IT = 0
    50     IT = IT+1
             IF(IT .LE. 30) GO TO 60
               FAIL = .TRUE.
               RETURN
    60       CONTINUE
             Z = A(L,L)
             R = X - Z
             S = Y - Z
             P = (R*S-W)/A(L+1,L) + A(L,L+1)
             Q = A(L+1,L+1) - Z - R - S
             R = A(L+2,L+1)
             S = ABS(P) + ABS(Q) + ABS(R)
             P = P/S
             Q = Q/S
             R = R/S
             CALL QRSTEP(A,V,P,Q,R,L,M,N,NA,NV)
           IF(ABS(A(M-1,M-2)) .GT. EPS*(ABS(A(M-1,M-1))+ABS(A(M-2,M-2))))
     1       GO TO 50
           A(M-1,M-2) = 0.
           RETURN
         CONTINUE
         END


         SUBROUTINE SPLIT(A,V,N,L,E1,E2,NA,NV)
C
         INTEGER L,N,NA,NV
         REAL A(NA,N),V(NV,N)
C
C        GIVEN THE UPPER HESSENBERG MATRIX A WITH A 2X2 BLOCK
C        STARTING AT A(L,L), SPLIT DETERMINES IF THE
C        CORRESPONDING EIGENVALUES ARE REAL OR COMPLEX. IF THEY
C        ARE REAL, A ROTATION IS DETERMINED THAT REDUCES THE
C        BLOCK TO UPPER TRIANGULAR FORM WITH THE EIGENVALUE
C        OF LARGEST ABSOLUTE VALUE APPEARING FIRST.  THE
C        ROTATION IS ACCUMULATED IN V.  THE EIGENVALUES (REAL
C        OR COMPLEX) ARE RETURNED IN E1 AND E2.  THE PARAMETERS
C        IN THE CALLING SEQUENCE ARE (STARRED PARAMETERS ARE
C        ALTERED BY THE SUBROUTINE)
C
C           *A        THE UPPER HESSENVERG MATRIX WHOSE 2X2
C                     BLOCK IS TO BE SPLIT.
C           *V        THE ARRAY IN WHICH THE SPLITTING TRANS-
C                     FORMATION IS TO BE ACCUMULATED.
C            N        THE ORDER OF THE MATRIX A.
C            L        THE POSITION OF THE 2X2 BLOCK.
C           *E1       ON RETURN IF THE EIGENVALUES ARE COMPLEX
C           *E2       E1 CONTAINS THEIR COMMON REAL PART AND
```

```
C                 E2 CONTAINS THE POSITIVE IMAGINARY PART.
C                 IF THE EIGENVALUES ARE REAL, E1 CONTAINS
C                 THE ONE LARGEST IN ABSOLUTE VALUE AND E2
C                 CONTAINS THE OTHER ONE.
C          NA     THE FIRST DIMENSION OF THE ARRAY A.
C          NV     THE FIRST DIMENSION OF THE ARRAY V.
C
C     INTERNAL VARIABLES
C
      INTEGER I,J,L1
      REAL P,Q,R,T,U,W,X,Y,Z
C
      X = A(L+1,L+1)
      Y = A(L,L)
      W = A(L,L+1)*A(L+1,L)
      P = (Y-X)/2.
      Q = P**2 + W
      IF(Q .GE. 0.) GO TO 5
C
C       COMPLEX EIGENVALUE.
C
        E1 = P + X
        E2 = SQRT(-Q)
        RETURN
    5 CONTINUE
C
C     TWO REAL EIGENVALUES.  SET UP TRANSFORMATION.
C
      Z = SQRT(Q)
      IF(P .LT. 0.) GO TO 10
        Z = P + Z
        GO TO 20
   10 CONTINUE
        Z = P - Z
   20 CONTINUE
      IF(Z .EQ. 0.) GO TO 30
        R = -W/Z
        GO TO 40
   30 CONTINUE
        R = 0.
   40 CONTINUE
      IF(ABS(X+Z) .GE. ABS(X+R)) Z = R
      Y = Y - X - Z
      X = -Z
      T = A(L,L+1)
      U = A(L+1,L)
      IF(ABS(Y)+ABS(U) .LE. ABS(T)+ABS(X)) GO TO 60
        Q = U
        P = Y
        GO TO 70
   60 CONTINUE
        Q = X
        P = T
   70 CONTINUE
      R = SQRT(P**2 + Q**2)
      IF(R .GT. 0.) GO TO 80
      E1 = A(L,L)
      E2 = A(L+1,L+1)
        A(L+1,L) = 0.
        RETURN
   80 CONTINUE
      P = P/R
      Q = Q/R
C
C     PREMULTIPLY.
C
```

```
      DO 90 J=L,N
        Z = A(L,J)
        A(L,J) = P*Z + Q*A(L+1,J)
        A(L+1,J) = P*A(L+1,J) - Q*Z
   90 CONTINUE
C
C     POSTMULTIPLY.
C
      L1 = L+1
      DO 100 I=1,L1
        Z = A(I,L)
        A(I,L) = P*Z + Q*A(I,L+1)
        A(I,L+1) = P*A(I,L+1) - Q*Z
  100 CONTINUE
C
C     ACCUMULATE THE TRANSFORMATION IN V.
C
      DO 110 I=1,N
        Z = V(I,L)
        V(I,L) = P*Z + Q*V(I,L+1)
        V(I,L+1) = P*V(I,L+1) - Q*Z
  110 CONTINUE
      A(L+1,L) = 0.
      E1 = A(L,L)
      E2 = A(L+1,L+1)
      RETURN
      END


      SUBROUTINE QRSTEP(A,V,P,Q,R,NL,NU,N,NA,NV)
C
      INTEGER N,NA,NL,NU,NV
      REAL A(NA,N),P,Q,R,V(NV,N)
C
C     QRSTEP PERFORMS ONE IMPLICIT QR STEP ON THE
C     UPPER HESSENBERG MATRIX A.  THE SHIFT IS DETERMINED
C     BY THE NUMBERS P,Q, AND R, AND THE STEP IS APPLIED TO
C     ROWS AND COLUMNS NL THROUGH NU.  THE TRANSFORMATIONS
C     ARE ACCUMULATED IN V.  THE PARAMETERS IN THE CALLING
C     SEQUENCE ARE (STARRED APRAMETERS ARE ALTERED BY THE
C     SUBROUTINE)
C
C        *A       THE UPPER HESSENBERG MATRIX ON WHICH THE
C                 QR STEP IS TO BE PERFORMED.
C        *V       THE ARRAY IN WHICH THE TRANSFORMATIONS
C                 ARE TO BE ACCUMULATED
C        *P       PARAMETERS THAT DETERMINE THE SHIFT.
C        *Q
C        *R
C         NL      THE LOWER LIMIT OF THE STEP.
C         NU      THE UPPER LIMIT OF THE STEP.
C         N       THE ORDER OF THE MATRIX A.
C         NA      THE FIRST DIMENSION OF THE ARRAY A.
C         NV      THE FIRST DIMENSION OF THE ARRAY V.
C
C     INTERNAL VARIABLES.
C
      INTEGER I,J,K,NL2,NL3,NUM1
      REAL S,X,Y,Z
      LOGICAL LAST
C
      NL2 = NL+2
      DO 10 I=NL2,NU
        A(I,I-2) = 0.
   10 CONTINUE
      IF(NL2 .EQ. NU) GO TO 30
```

```
      NL3 = NL+3
      DO 20 I=NL3,NU
        A(I,I-3) = 0.
   20 CONTINUE
   30 CONTINUE
      NUM1 = NU-1
      DO 130 K=NL,NUM1
C
C       DETERMINE THE TRANSFORMATION.
C
        LAST = K .EQ. NUM1
        IF(K .EQ. NL) GO TO 40
          P = A(K,K-1)
          Q = A(K+1,K-1)
          R = 0.
          IF(.NOT.LAST) R = A(K+2,K-1)
          X = ABS(P) + ABS(Q) + ABS(R)
          IF(X .EQ. 0.) GO TO 130
          P = P/X
          Q = Q/X
          R = R/X
   40   CONTINUE
        S = SQRT(P**2 + Q**2 + R**2)
        IF(P .LT. 0.) S = -S
        IF(K .EQ. NL) GO TO 50
          A(K,K-1) = -S*X
          GO TO 60
   50   CONTINUE
          IF(NL .NE. 1) A(K,K-1) = -A(K,K-1)
   60   CONTINUE
        P = P + S
        X = P/S
        Y = Q/S
        Z = R/S
        Q = Q/P
        R = R/P
C
C       PREMULTIPLY.
C
        DO 80 J=K,N
          P = A(K,J) + Q*A(K+1,J)
          IF(LAST) GO TO 70
            P = P + R*A(K+2,J)
            A(K+2,J) = A(K+2,J) - P*Z
   70     CONTINUE
          A(K+1,J) = A(K+1,J) - P*Y
          A(K,J) = A(K,J) - P*X
   80   CONTINUE
C
C       POSTMULTIPLY.
C
        J = MINO(K+3,NU)
        DO 100 I=1,J
          P = X*A(I,K) + Y*A(I,K+1)
          IF(LAST) GO TO 90
            P = P + Z*A(I,K+2)
            A(I,K+2) = A(I,K+2) - P*R
   90     CONTINUE
          A(I,K+1) = A(I,K+1) - P*Q
          A(I,K) = A(I,K) - P
  100   CONTINUE
C
C       ACCUMULATE THE TRANSFORMATION IN V.
C
        DO 120 I=1,N
          P = X*V(I,K) + Y*V(I,K+1)
```

```
      IF(LAST) GO TO 110
        P = P + Z*V(I,K+2)
        V(I,K+2) = V(I,K+2) - P*R
110   CONTINUE
      V(I,K+1) = V(I,K+1) - P*Q
      V(I,K) = V(I,K) - P
120 CONTINUE
130 CONTINUE
    RETURN
    END


    SUBROUTINE ORTHES(NM,N,LOW,IGH,A,ORT)
    INTEGER I,J,M,N,II,JJ,LA,MP,NM,IGH,KP1,LOW
    REAL A(NM,N),ORT(IGH)
    REAL F,G,H,SCALE
    LA = IGH - 1
    KP1 = LOW + 1
    IF(LA .LT. KP1) GO TO 200
    DO 180 M=KP1,LA
      H = 0.
      ORT(M) = 0.
      SCALE = 0.
      DO 90 I=1,IGH
 90   SCALE = SCALE + ABS(A(I,M-1))
      IF(SCALE .EQ. 0.) GO TO 180
      MP = M + IGH
      DO 100 II=M,IGH
        I = MP - II
        ORT(I) = A(I,M-1)/SCALE
        H = H + ORT(I)*ORT(I)
100   CONTINUE
      G = -SIGN(SQRT(H),ORT(M))
      H = H- ORT(M)*G
      ORT(M) = ORT(M) - G
      DO 130 J=M,N
        F = 0.
        DO 110 II=M,IGH
          I = MP - II
          F = F + ORT(I)*A(I,J)
110     CONTINUE
        F = F/H
        DO 120 I=M,IGH
          A(I,J) = A(I,J) - F*ORT(I)
120     CONTINUE
130   CONTINUE
      DO 160 I=1,IGH
        F = 0.
        DO 140 JJ=M,IGH
          J = MP - JJ
          F = F + ORT(J)*A(I,J)
140     CONTINUE
        F = F/H
        DO 150 J=M,IGH
          A(I,J) = A(I,J) - F*ORT(J)
150     CONTINUE
160   CONTINUE
      ORT(M) = SCALE*ORT(M)
      A(M,M-1) = SCALE*G
180 CONTINUE
200 RETURN
    END


    SUBROUTINE ORTRAN(NM,N,LOW,IGH,A,ORT,Z)
    INTEGER I,J,N,KL,MM,MP,NM,IGH,LOW,MP1
```

```
      REAL A(NM,IGH),ORT(IGH),Z(NM,N)
      REAL G,H
      DO 80 I=1,N
        DO 60 J=1,N
          Z(I,J) = 0.
   60   CONTINUE
        Z(I,I) = 1.
   80 CONTINUE
      KL = IGH - LOW - 1
      IF(KL .LT. 1) GO TO 200
      DO 140 MM=1,KL
        MP = IGH - MM
        H = A(MP,MP-1)*ORT(MP)
        IF(H .EQ. 0.) GO TO 140
        MP1 = MP+1
        DO 100 I=MP1,IGH
          ORT(I) = A(I,MP-1)
  100   CONTINUE
        DO 130 J=MP,IGH
          G = 0.
          DO 110 I=MP,IGH
            G = G + ORT(I)*Z(I,J)
  110     CONTINUE
          G = G/H
          DO 120 I=MP,IGH
            Z(I,J) = Z(I,J) + G*ORT(I)
  120     CONTINUE
  130   CONTINUE
  140 CONTINUE
  200 RETURN
      END
```